Electronic Notes in Theoretical Computer Science 74 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume74.html 18 pages

On the construction of a new generalization of Runge–Kutta methods

Mark Sofroniou [1]

Department of Research and Development,
Wolfram Research Inc., Champaign, Illinois, USA

Giulia Spaletta [2,3]

Department of Mathematics, Bologna University,
Piazza Porta S.Donato 5, 40127 Bologna, Italy

Abstract

We give an overview of the construction of algebraic conditions for determining the order of Runge–Kutta methods and describe a novel extension for numerically solving systems of differential equations. The new schemes, called Elementary Differential Runge–Kutta methods, include as a subset Runge–Kutta methods, Taylor series methods, Multiderivative Runge–Kutta methods. We outline how order conditions have been constructed for the new schemes using B-series and their composition and give details relating to a *Mathematica* implementation.

Keywords. Elementary differentials; B-series; Generalized Runge–Kutta methods; geometric numerical integration.

AMS. 65L05, 05C05

1 Introduction.

In recent years there has been a growth of interest in the field of geometric numerical integration, as evidenced by the appearance of the recent monograph [8]. The essential idea is that if a differential system has a certain structural property, then the numerical method should preserve this structure as closely as possible. This results in improved qualitative behavior, which is particularly evident in long time simulations.

[1] Email: marks@wolfram.com
[2] Email: giulia@dm.unibo.it
[3] Work partially supported by the University of Bologna, funds for selected research topics.

Certain types of numerical methods that use derivatives of the vector field cannot always reproduce the same qualitative behavior as methods involving just the vector field [9]. To overcome this limitation, a new class of methods is introduced here. Details relating to the automation of the new theory are given together with explanatory examples.

This article is organized as follows. Section 2 contains a survey of the modern graph theory based formalism for the generation of order conditions of Runge–Kutta methods that is based on trees. Then we recall multiderivative Runge–Kutta methods along with the more general concept of B-series. In Section 3 a new class of numerical methods is introduced and the connection with a graph approach is described, based on a more specific type of tree. In Section 4 order conditions for the new numerical schemes are derived along with an example method. In Section 5 a numerical example is given to illustrate the advantageous qualitative behavior of the new schemes.

2 Background.

Consider the solution of an initial value problem, given in autonomous form for simplicity and without loss of generality:

$$\mathbf{y}'(t) = \mathbf{f}(\mathbf{y}(t)), \quad \mathbf{y}(0) = \mathbf{y}_0, \quad \mathbf{y} \in \mathbb{R}^d, \quad \mathbf{f} : \mathbb{R}^d \mapsto \mathbb{R}^d. \tag{1}$$

To obtain approximate solutions, the continuous system (1) is replaced by a discrete map $\psi_{\mathbf{f},h}$, dependent upon the system under consideration, the numerical solver used and a step-size h. The time-evolution of (1) is then modeled by iterating $\psi_{\mathbf{f},h}$ on the initial conditions.

A one-step numerical method has order p if, for all sufficiently regular problems (1), the local error after one step satisfies $\mathbf{y}_1 - \mathbf{y}(h) = O(h^{p+1})$ as $h \to 0$ [8]. To check the order of a one-step method, the Taylor series expansion of both $\mathbf{y}(h)$ and $\mathbf{y}_1$ have to be computed around $h = 0$. This leads to algebraic order conditions to be imposed on the method coefficients.

There is a one-to-one correspondence between *rooted trees* and the *elementary differentials* appearing in the Taylor expansion. This relationship is illustrated in Sections 2.2 and 2.4 (see [3, theorem 307B] for proof). Trees were used in this context by Merson [15] and subsequently developed into a major theory by Butcher and later by Hairer and Wanner (see [3,6,7,8] for a summary). The process can be automated using computer algebra and the *Mathematica* package **Butcher.m** is the first to implement the tree formalism [18]. Techniques for constructing specialized methods can readily be built upon the functionality [11,19]. In the work outlined here, we will use this package as a starting point for our implementation.

2.1 Runge–Kutta methods

We now recall the definition of Runge–Kutta (RK) methods, using a form that facilitates composition using B-series in Definition 2.2.

The s-stage RK method for the differential system (1) is:

$$\begin{aligned} \mathbf{Y}_i &= \mathbf{y}_n + h \sum_{j=1}^{s} a_{i,j}\, \mathbf{f}(\mathbf{Y}_j), \quad i = 1, \ldots, s, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h \sum_{i=1}^{s} b_i\, \mathbf{f}(\mathbf{Y}_i) \end{aligned} \tag{2}$$

and is of order p if its Taylor series expansion agrees with the exact solution up to and including terms in h^p, but *not all* terms in h^{p+1}.

The parameters $a_{i,j}$, b_i and c_i are chosen such that $\mathbf{y}_n$ represents an approximation to the Taylor series expansion of the solution to some order in the time step h. It has become standard practice to represent the free parameters of RK methods using a *Butcher tableau* [3]:

$$\begin{array}{c|ccc} c_1 & a_{1,1} & \cdots & a_{1,s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & \cdots & a_{s,s} \\ \hline & b_1 & \cdots & b_s \end{array} \tag{3}$$

The *row-sum conditions* $c_i = \sum_{j=1}^{s} a_{i,j},\ i = 1, 2, \ldots, s$, are usually assumed to hold and appear naturally in the derivation of high order methods [6]. If the condition $a_{i,j} = 0,\ 1 \leq i \leq j \leq s,$ holds, then the method is said to be *explicit* and the intermediate stages $\mathbf{Y}_i$ in (2) can be computed sequentially.

2.2 Representation of trees.

A tree constitutes a set of vertices and edges and is a graph with no cycles. A rooted tree is a tree where the root has been highlighted; graphically this is emphasized by encircling the root [10]. All trees may be constructed in terms of the trivial tree (tree with one vertex) τ.

Since we are concerned with unordered trees, where the order of the branching is unimportant, exponentiation can be used to denote multiple branching of a subtree, k-fold branching of a subtree u being represented by u^k. Grafting is the process of joining a tree to a new vertex (the root) via an edge; l-fold grafting onto a new root is denoted by $[_l u]_l$, the subscript being omitted if $l = 1$. Butcher adopts the convention of balanced subscripts [3], so that for example $[_2\tau][_2\tau]_3$ is instead represented as $[[\tau][_2\tau]_2]$.

In *Mathematica*, each tree is represented by a *functional form*, using the symbol **f**. Table 1 illustrates the relationship between the two representations, while an example tree is depicted in Figure 1.

2.3 Graph definitions

Let $\mathcal{T}$ denote the set of rooted trees and consider a tree:

$$u = [u_1^{m_1}, \ldots, u_n^{m_n}]\,, \tag{4}$$

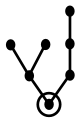

Fig. 1. The rooted tree $[[\tau^2][{}_2\tau]_2]$, represented in *Mathematica* as $\mathbf{f}[\mathbf{f}[\mathbf{f^2}]\ \mathbf{f}[\mathbf{f}]]$.

τ	$[\tau]$	$[{}_2\tau]_2$	$[\tau^2]$	$[{}_3\tau]_3$	$[{}_2\tau^2]_2$	$[\tau[\tau]]$	$[\tau^3]$
$\mathbf{f}$	$\mathbf{f}[\mathbf{f}]$	$\mathbf{f}[\mathbf{f}[\mathbf{f}]]$	$\mathbf{f}[\mathbf{f^2}]$	$\mathbf{f}[\mathbf{f}[\mathbf{f}[\mathbf{f}]]]$	$\mathbf{f}[\mathbf{f}[\mathbf{f^2}]]$	$\mathbf{f}[\mathbf{f}\,\mathbf{f}[\mathbf{f}]]$	$\mathbf{f}[\mathbf{f^3}]$

Table 1
Illustration of the correspondence between Butcher's notation (above) for the representation of trees and the functional form used in *Mathematica* (below).

where m_i denotes the multiplicity, or number of occurrences, of the subtree u_i. We will need the functions r (the order), σ (the symmetry), γ (the density) and α (the number of monotonic labellings), $r,\ \sigma,\ \gamma,\ \alpha :\ \mathcal{T} \mapsto \mathbb{N}$ which are defined recursively for the tree in (4) as:

$$
\begin{aligned}
r(u) &= 1 + \sum_{i=1}^{n} m_i\, r(u_i), \\
\sigma(u) &= \prod_{i=1}^{n} m_i!\, \sigma(u_i)^{m_i}\,, \\
\gamma(u) &= r(u) \prod_{i=1}^{n} \gamma(u_i)^{m_i}\,, \\
\alpha(u) &= \begin{pmatrix} r(u)-1 \\ m_1\, r(u_1), \ldots, m_n\, r(u_n) \end{pmatrix} \prod_{i=1}^{n} \left(\frac{(m_i\, r(u_i))!}{m_i!} \right) \left(\frac{\alpha(u_i)}{r(u_i)!} \right)^{m_i}\,,
\end{aligned}
\tag{5}
$$

with $r(\tau) = \alpha(\tau) = \gamma(\tau) = \sigma(\tau) = 1$ and where the multinomial coefficient appears in the definition of α. The following relationship holds:

$$
\alpha(u)\, \sigma(u)\, \gamma(u) = r(u)!\,. \tag{6}
$$

The reciprocal of the density $\gamma(u)$ of a rooted tree u occurs in the right hand side of the order conditions for RK methods (see (9)). Table 2 summarizes the values of the quantities defined in (5) for the rooted trees through order 4.

2.4 Elementary differentials

The formation of the elementary differential (or Frechet derivative) associated with a rooted tree u is now illustrated. Given $u \in \mathcal{T}$ of order $r(u)$, associate a distinct label $1, \ldots, r(u)$, with each vertex, to obtain a rooted labelled tree. For each vertex write $\mathbf{f}$ with superscript i_k where k is the label of the vertex. Add subscripts i_l corresponding to the labelling of *each* descendant l *unless* the vertex is terminal (has no descendants). Summing the product of these terms over non-rooted indices gives the i_mth component of the required vector

Name	τ	τ_2	$\tau_{3,1}$	$\tau_{3,2}$	$\tau_{4,1}$	$\tau_{4,2}$	$\tau_{4,3}$	$\tau_{4,4}$
Representation	τ	$[\tau]$	$[{}_2\tau]_2$	$[\tau^2]$	$[{}_3\tau]_3$	$[{}_2\tau^2]_2$	$[\tau[\tau]]$	$[\tau^3]$
Order r	1	2	3	3	4	4	4	4
Density γ	1	2	6	3	24	12	8	4
Symmetry σ	1	1	1	2	1	2	1	6
Monotonic labellings α	1	1	1	1	1	1	3	1

Table 2
The nomenclature and some numbers related to the first few rooted trees.

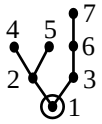

Fig. 2. A rooted labelled tree corresponding to the rooted tree $[[\tau^2][{}_2\tau]_2]$.

derivative, where m is the label of the root. For example, consider the tree depicted in Figure 1. Attach labels to obtain the tree depicted in Figure 2. The corresponding vector component is:

$$\mathbf{f}^{i_1}_{i_2 i_3}\,\mathbf{f}^{i_2}_{i_4 i_5}\,\mathbf{f}^{i_4}\,\mathbf{f}^{i_5}\,\mathbf{f}^{i_3}_{i_6}\,\mathbf{f}^{i_6}_{i_7}\,\mathbf{f}^{i_7} \tag{7}$$

where superscripts denote components and subscripts denote partial derivatives. The summation convention is assumed (implicit summation over all superscripts $i_2, \ldots, i_{r(u)}$ not associated with the root). The elementary differential is formed by summing the components (7) over i_1.

As an example, consider the following third order total derivative:

$$\mathbf{y}^{(3)} = \mathbf{f}^{(2)}(\mathbf{f}, \mathbf{f}) + \mathbf{f}^{(1)}(\mathbf{f}^{(1)}(\mathbf{f}))\,. \tag{8}$$

Notice that the relation involves the two elementary differentials associated with the third order rooted trees of Table 1. When the differential system (1) is scalar valued, then (8) can be written in the (perhaps more familiar) form:

$$y^{(3)} = f''\,f^2 + (f')^2\,f\,.$$

The notation f_y is also commonly used to indicate the dependence of the derivative explicitly.

The following definition relates elementary differentials to trees.

Definition 2.1 For the tree u in (4), the elementary differential is a mapping $F(u) : \mathbb{R}^d \mapsto \mathbb{R}^d$ defined recursively by

$$F(u)(\mathbf{y}) = \mathbf{f}^{(m)}(\mathbf{y})(\underbrace{F(u_1)(\mathbf{y}), \ldots}_{m_1},\ \ldots,\ \underbrace{F(u_n)(\mathbf{y})), \ldots}_{m_n})\,,$$

with $F(\tau)(\mathbf{y}) = \mathbf{f}(\mathbf{y})$ and $m = \sum_{i=1}^{n} m_i$.

The number of occurrences of the elementary differential associated with the rooted tree u, appearing in a Taylor expansion of the exact solution, is given by the number of monotonic labellings $\alpha(u)$ (see (11)).

2.5 Elementary weights

The left hand side of an order condition for an RK method associated with $u \in \mathcal{T}$ is formed via the *elementary weight function* $\Phi(u)$ which is a polynomial in the method coefficients. Butcher proved that necessary and sufficient conditions for a pth order RK method are given by [3]:

$$\Phi(u) = \frac{1}{\gamma(u)}, \quad \forall r(u) \leq p\,. \tag{9}$$

Additional relations, such as stability properties of the method or Butcher's simplifying assumptions, may give rise to a reduction in the number of trees which need to be considered [19].

An algorithm for evaluating the elementary weight function is outlined here, assuming the notation used for the Runge–Kutta method coefficients and the row-sum conditions (see Section 2.1). Consider the rooted labelled tree of Figure 2. To the root with label j associate the quantity b_{i_j}. Consider each vertex pair adjoined by an edge with labels k, l, ordered so that l denotes the label of the vertex farthest from the root. To each such pair, associate the quantity $a_{i_k i_l}$, *unless* the vertex with label l is terminal, when the term c_{i_k} is used instead. The elementary weight function is then formed by summing the product of these terms over the indices from 1 to the number of stages s. It is not difficult to see that the summation is in fact independent of the labelling of the tree.

In **Butcher.m**, $\Phi(u)$ is calculated by recursing through all levels of a tree, starting at the root and generating the required indices along the way. Bookkeeping stores the index at the previous level. A finishing touch ensures that the i_k are displayed as consecutive letters $i, j, k, \ldots$ etc.

An alternative strategy for deriving the elementary weight function $\Phi(u)$ has also been implemented using a tensor formulation. Advantages are that the tensor form is independent of the number of stages of the method, it is more compact (consumes less memory) and much more efficient than the index notation described above. Summation over repeated indices in the elementary weight function is recast in terms of vector and matrix operations. To illustrate the correspondence, $\Phi(u)$ is compared in Table 3 for trees of order $r \leq 4$ using tensor notation and Butcher's index notation. The *Mathematica* representation of the tensor notation is also given. In the tensor notation, $\mathbf{a} = \mathbf{A} = [a_{i,j}]$ denotes an $s \times s$ matrix and $\mathbf{b} = [b_i]$, $\mathbf{c} = [c_i]$ denote s-dimensional vectors. In addition, the s-dimensional vector $\mathbf{e} = [1, 1, \ldots, 1]^T$ is introduced along with a diagonal matrix $\mathbf{C} = diag(c_1, \ldots, c_s)$. The operation of exponentiation of a vector is interpreted componentwise.

Functional tree	Tensor Φ	*Mathematica*	Index Φ
$\mathbf{f}$	$\mathbf{b}^T\,\mathbf{e}$	**b.e**	$\sum_i b_i$
$\mathbf{f}[\mathbf{f}]$	$\mathbf{b}^T\,\mathbf{c}$	**b.c**	$\sum_i b_i c_i$
$\mathbf{f}[\mathbf{f}[\mathbf{f}]]$	$\mathbf{b}^T\,\mathbf{A}\,\mathbf{c}$	**b.a.c**	$\sum_{i,j} b_i a_{i,j} c_j$
$\mathbf{f}[\mathbf{f^2}]$	$\mathbf{b}^T\,\mathbf{c}^2$	**b.**($\mathbf{c}^2$)	$\sum_i b_i c_i^2$
$\mathbf{f}[\mathbf{f}[\mathbf{f}[\mathbf{f}]]]$	$\mathbf{b}^T\,\mathbf{A}\,\mathbf{A}\,\mathbf{c}$	**b.a.a.c**	$\sum_{i,j,k} b_i a_{i,j} a_{j,k} c_k$
$\mathbf{f}[\mathbf{f}[\mathbf{f^2}]]$	$\mathbf{b}^T\,\mathbf{A}\,\mathbf{c}^2$	**b.a.**($\mathbf{c}^2$)	$\sum_{i,j} b_i a_{i,j} c_j^2$
$\mathbf{f}[\mathbf{f}\,\mathbf{f}[\mathbf{f}]]$	$\mathbf{b}^T\,\mathbf{C}\,\mathbf{A}\,\mathbf{c}$	**b.**($\mathbf{c} * \mathbf{a.c}$)	$\sum_{i,j} b_i c_i a_{i,j} c_j$
$\mathbf{f}[\mathbf{f^3}]$	$\mathbf{b}^T\,\mathbf{c}^3$	**b.**($\mathbf{c}^3$)	$\sum_i b_i c_i^3$

Table 3
Different notations used to represent the elementary weight function Φ.

Evaluation of the elementary weight function for a set of trees is not necessarily an independent process. For example, the third, fifth and seventh trees depicted in Table 3 share the same operation **a.c**; this result can be stored and evaluated only once. Another advantage of the tensor representation is that it is independent of the particular method (explicit, implicit, etc.).

2.6 Multiderivative Runge–Kutta methods

Multi–Derivative Runge–Kutta (MDRK) methods generalize RK methods by including contributions from total derivatives of the vector field [12].

Let $a_{i,j}^{(k)}$, $b_i^{(k)}$, $i, j = 1, \ldots, s,\ k = 1, \ldots, q,$ be real coefficients. The s-stage q-derivative RK method for the differential system (1) is given by:

$$\begin{aligned} \mathbf{Y}_i &= \mathbf{y}_n + \sum_{k=1}^{q} \frac{h^k}{k!} \sum_{j=1}^{s} a_{i,j}^{(k)}\, D^k(\mathbf{Y}_j), \quad i = 1, \ldots, s\,, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \sum_{k=1}^{q} \frac{h^k}{k!} \sum_{i=1}^{s} b_i^{(k)}\, D^k(\mathbf{Y}_i). \end{aligned} \tag{10}$$

D denotes the differential operator, D^k the total derivative of order k and $D^1 = \mathbf{f}$. MDRK methods involve a table of free parameters that are associated with *each* total derivative. An important special case of MDRK methods

are Taylor series methods:

$$a_{i,j}^{(k)} = 0,\ i,j = 1,\dots s,\ k = 1,\dots q\,.$$

Let $\mathcal{T}_k$ denote the set of rooted trees u of order k, then the k-th total derivative is a linear combination, with coefficients $\alpha(u)$, of elementary differentials $F(u)$ [3]:

$$D^k = \sum_{u\in\mathcal{T}_k} \alpha(u)\,F(u) \tag{11}$$

The linear combination (11) with $k = 3$ is given in (8).

2.7 B-Series

The concept of B-series gives insight into the behavior of numerical methods and allows extensions to more general classes of methods. The main results of the theory of B-series began with the work of Butcher [2] and were subsequently generalized by Hairer and Wanner [5].

Let $\mathbf{a} : \mathcal{T} \cup \{\emptyset\} \mapsto \mathbb{R}$ be a sequence of real coefficients defined for all trees (not be confused with the coefficients $\mathbf{a}$ used in the notation for RK methods). Then the following is a B-series [4]:

$$B(\mathbf{a},\mathbf{y}) = \mathbf{a}(\emptyset)\,\mathbf{y} + \sum_{u\in\mathcal{T}} \frac{h^{r(u)}}{\sigma(u)}\,\mathbf{a}(u)\,F(u)(\mathbf{y})\,. \tag{12}$$

The exact solution of (1) is a B-series. The numerical solution of a number of numerical schemes, including RK methods and MDRK methods, are also B-series.

2.8 Composition of B-Series

The composition of two B-series can be represented as $B(\mathbf{b}, B(\mathbf{a},\mathbf{y})) = B(\mathbf{ab},\mathbf{y})$ in terms of the product of the coefficients, according to the following definition.

Definition 2.2 Let $\mathbf{a},\mathbf{b} : \mathcal{T}\cup\{\emptyset\} \mapsto \mathbb{R}$ be two sequences with $\mathbf{a}(\emptyset) = 1$. Then for a tree $u \in \mathcal{T}$ define the composition:

$$\mathbf{ab}(u) = \sum_{v<u} \mathbf{b}(v) \prod_{w\in u\setminus v} \mathbf{a}(w)\,,$$

where $v < u$ denotes that v is a subtree of u sharing the same root and $u \setminus v$ is the set of trees remaining after deleting the subtree v from u at the root.

Figure 3 gives an example of the process outlined in Definition 2.2. The composition formula for B-series gives a recursive procedure for deriving order conditions. For RK methods Definition 2.2 should be applied repeatedly by considering the cuttings of a tree u induced by the tree of order one, τ. This

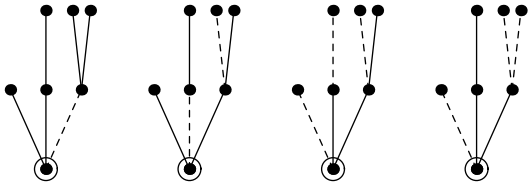

Fig. 3. Cuttings of the tree $u = [\tau[\tau][\tau^2]]$ induced by a tree $v = [\tau[\tau]]$ together with the remainder sets (dashed lines).

gives rise to the recursive procedure for determining the elementary weight outlined in Section 2.5 (see [8] for more details).

3 New schemes

The goal of the remainder of this article is to show that there exist methods which use different combinations of elementary differentials resulting in additional degrees of freedom when compared with MDRK schemes. This extra freedom can be used to attain improved qualitative behavior.

3.1 Elementary Differential Runge–Kutta methods

Making use of elementary differentials to compute approximations at the internal stages, we define an *Elementary Differential Runge–Kutta* method as:

$$\begin{aligned} \mathbf{Y}_i &= \mathbf{y}_n + \sum_{u\in\mathcal{U}} \frac{h^{r(u)}}{r(u)!} \sum_{j=1}^{s} a(u)_{i,j}\, F(u)(\mathbf{Y}_j), \quad i = 1,\ldots,s \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \sum_{u\in\mathcal{U}} \frac{h^{r(u)}}{r(u)!} \sum_{i=1}^{s} b(u)_i\, F(u)(\mathbf{Y}_i); \end{aligned} \tag{13}$$

where $\mathcal{U}$ denotes a finite set of trees $\mathcal{U} \subset \mathcal{T}$. EDRK methods constitute a subset of B-series methods and contain, in turn, MDRK methods as a subset. EDRK schemes have one table of free parameters for *each* elementary differential.

3.2 Connection with P-trees

It is sometimes advantageous to consider solving differential systems that are partitioned and to solve each system with a suitable numerical method (for example a stiff and a non-stiff solver). The analysis of partitioned systems of differential equations can be accomplished using P-trees, or bicolor trees with two types of non-root vertices (see [6] for a summary). Here we will consider an application of P-trees in a slightly different context.

Order conditions for EDRK methods can be derived using the composition formula for B-series outlined in Section 2.8. However, the situation is somewhat more complicated for EDRK methods than it was for RK methods,

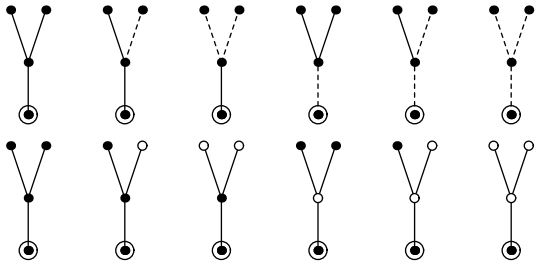

Fig. 4. Decomposition of the tree $[{}_2\tau^2]_2$ (above) and the associated P-trees (below).

Order	1	2	3	4	5	6	7	8	9	10
Order conditions	1	1	2	4	9	20	48	115	286	719
Distinct components	1	2	7	26	107	458	2058	9498	44947	216598

Table 4
The number of order conditions (rooted trees) and the number of decompositions (P-trees) at various orders.

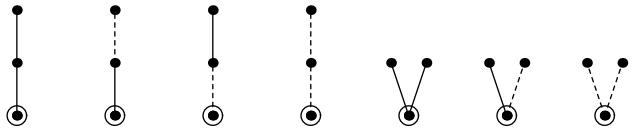

Fig. 5. Decomposition of third order trees $[{}_2\tau]_2$ (left) and $[\tau^2]$ (right).

since we need to consider the remainder sets induced by all possible subtrees of a tree u.

As will be illustrated in Section 4.1, the number of distinct terms in each order condition for EDRK methods depends on the number of ways that a tree can be decomposed into subtrees. The number of distinct decompositions of trees $u \in \mathcal{T}_k$ corresponds to the number of P-trees of order k. The decomposition of the tree $[{}_2\tau^2]_2$ into subtrees is displayed graphically in Figure 4 together with the associated P-trees. The second and fifth trees turn up twice. The number of P-trees at each order is given in Table 4. As a further example, consider the trees of order three. There are two rooted trees $[{}_2\tau]_2$ and $[\tau^2]$ which have four decompositions (associated P-trees) and three decompositions respectively, as shown in Figure 5.

3.3 Generation of P-trees

Now that we have outlined the role of P-trees in the B-series composition formula for EDRK methods, we require an algorithmic process for actually

generating them. Given a particular rooted tree our goal is to generate the associated P-trees. **Butcher.m** is used to generate rooted trees in a functional form, in terms of the single symbol **f**. The essence of our implementation is that each rooted tree should be traversed recursively and each non-root vertex is then represented in two ways with either the symbol **f** or the symbol **g**. Details of this process are now summarized, together with some optimizations required for efficient implementation. Before continuing it is useful to establish lower and upper bounds on the number of P-trees associated with a given rooted tree.

For straight trees (where each vertex has only one descendant) and trees with unit symmetry, $\sigma(u) = 1$, there are $2^{r(u)-1}$ associated P-trees (see the leftmost trees in Figure 5). Thus we simply need to generate all possible combinations of non-root vertices alternately using the symbol **f** or **g**. Sorting is then used to ensure that the sub-trees are generated in a canonical form.

For remaining trees with $\sigma(u) > 1$, the representation (4) necessarily involves a subtree with $u_i^{m_i},\ m_i > 1$, so that detection of this situation in a recursive traversal is straightforward. There are a number of possible implementation strategies.

To derive a lower bound on the number of P-trees arising from a rooted tree, consider the bushy trees $[\tau^m],\ m > 1$, where every non-root vertex is terminal. There are exactly $m+1$ associated P-trees and they can be efficiently generated using a binomial expansion. For example, there are $m+1$ terms for $(f+g)^m$, namely $f^m, f\,g^{m-1}, \ldots, g^m$. See the rightmost trees in Figure 5 for illustration.

Powers of more general subtrees $u^m,\ m > 1,\ u \neq \tau$ can be generated by considering a polynomial of k distinct terms, where k is the number of P-trees associated with the subtree u. These powers can be generated directly using a multinomial expansion (see [1, pg. 283]):

$$(x_1 + x_2 + \cdots + x_k)^m = \sum_{m_1+m_2+\cdots+m_k=m} \binom{m}{m_1, m_2, \ldots, m_k} x_1^{m_1} x_2^{m_2} \cdots x_k^{m_k}\,.$$

The summation constraint can be handled by considering the partitions of integers; for $m = 3$ it suffices to consider all possible combinations of $\{m_1, m_2, m_3\}$ which take the values $\{3,0,0\}$, $\{2,1,0\}$, $\{1,1,1\}$.

Despite the existence of this direct multinomial expansion, we found that there are more efficient ways of proceeding. One approach, which also avoids generating duplicates, is to recursively decompose the multinomial into binomials. However, we found that it is more efficient to handle powers using a binary power decomposition and simply sort the results and collect duplicates. Intermediary results are hashed for efficient recall. We have chosen the left-right binary decomposition since in general it generates fewer intermediary duplicates than the right-left form (see [13] for a discussion of the different product sequences involved). Often, as is the case in Section 4.1,

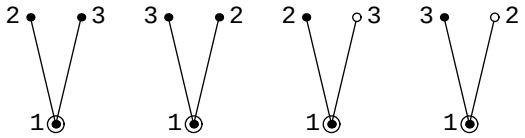

Fig. 6. Monotonic labellings of the third order rooted tree $[\tau^2]$ (left) and an associated P-tree (right).

we are interested in simultaneously generating P-trees from a given forest of trees. An advantage of the binary power decomposition approach is that it allows intermediary results to be shared across trees.

The next result that we require is a way of counting the number of times a P-tree v turns up in the expansion of a rooted tree u into P-trees. In order to accomplish this the quantities defined in (5) can be extended in a straightforward way to P-trees (see [8]). Now consider the identity (6). Since $r(u) = r(v)$ and $\gamma(u) = \gamma(v)$ then the following integer relationship holds:

$$\frac{\alpha(v)}{\alpha(u)} = \frac{\sigma(u)}{\sigma(v)}\,. \tag{14}$$

Formula (14) counts the number of repetitions of a P-tree v that is associated with a given tree u.

Figure 6 depicts an example of the computation of $\alpha(u)$ and $\alpha(v)$ for a third order tree $u = [\tau^2]$ and an associated P-tree v. The two labelled trees associated with u are topologically equivalent, since we are dealing with unordered trees, so that $\alpha(u) = 1$. However the two labelled trees associated with v are topologically distinct so that $\alpha(v) = 2$. Therefore from (14) it follows that the P-tree v turns up twice in the expansion of u into P-trees.

Computationally it is generally more efficient to use $\sigma(u)$ for the computation in (14) instead of $\alpha(u)$ (and the symmetries are needed in any case in the computation of the order conditions in Section 4.1).

4 Derivation of EDRK methods

Now that a process for the generation of P-trees associated with a rooted tree has been outlined and formula (14) for the number of occurrences of each P-tree has been derived, we are in a position to determine the order conditions for EDRK methods. All that remains is to define a function to compute the elementary weight function for EDRK methods. This entails generating the tree dependent coefficients by recursive traversal of all P-trees associated with each rooted tree. During the recursion of each P-tree it is necessary to determine the tree quotient and remainder sets appearing in Definition 2.2, so that each coefficient can be labelled with its associated subtree. In order to match the definition of an EDRK method (13) with Definition 2.2 we need to recursively rescale the coefficients by a factor of $\sigma(u)/r(u)!$ for all subtrees u.

4.1 EDRK order conditions

Table 5 gives the order conditions through order four for EDRK methods using a tensor notation. In contrast to RK methods, each coefficient is associated with a tree (and hence an associated elementary differential). Consider the number of algebraic terms at order three in Table 4. Of the seven distinct components, four appear in the order condition associated with $\tau_{3,1}$ in Table 5 and three appear in the other condition for $\tau_{3,2}$ (see Figure 5 for illustration).

Order conditions for standard RK methods are recovered from Table 5 by considering only coefficients involving the tree τ (the last component on the left hand side of each order condition).

Comparing the new order conditions in Table 5 with those for MDRK methods (which constitute a subset) reveals a typographical error in [6, Section II.13, Exercise 3] where the term $\sum_i b_i^{(3)}$ has been omitted.

4.2 Example method

As an example consider the order conditions for one stage implicit EDRK schemes of order four which involve the trees τ, $\tau_{3,1}$ and $\tau_{3,2}$. This yields the following algebraic equations:

$$
\begin{aligned}
b(\tau)_1 &= 1\,, \\
b(\tau)_1\, a(\tau)_{1,1} &= \tfrac{1}{2}\,, \\
\tfrac{1}{6}\, b(\tau_{3,1})_1 + b(\tau)_1\, a(\tau)_{1,1}^2 &= \tfrac{1}{6}\,, \\
\tfrac{1}{3}\, b(\tau_{3,2})_1 + b(\tau)_1\, a(\tau)_{1,1}^2 &= \tfrac{1}{3}\,, \\
\tfrac{1}{6}\, b(\tau_{3,1})_1\, a(\tau)_{1,1} + \tfrac{1}{6}\, b(\tau)_1\, a(\tau_{3,1})_{1,1} + b(\tau)_1\, a(\tau)_{1,1}^3 &= \tfrac{1}{24}\,, \\
\tfrac{1}{3}\, b(\tau_{3,1})_1\, a(\tau)_{1,1} + \tfrac{1}{3}\, b(\tau)_1\, a(\tau_{3,2})_{1,1} + b(\tau)_1\, a(\tau)_{1,1}^3 &= \tfrac{1}{12}\,, \\
\tfrac{1}{6}\, b(\tau_{3,1})_1\, a(\tau)_{1,1} + \tfrac{1}{3}\; b(\tau_{3,2})_1\, a(\tau)_{1,1} + b(\tau)_1\, a(\tau)_{1,1}^3 &= \tfrac{1}{8}\,, \\
b(\tau_{3,2})_1\, a(\tau)_{1,1} + b(\tau)_1\, a(\tau)_{1,1}^3 &= \tfrac{1}{4}\,.
\end{aligned}
\tag{15}
$$

It is not difficult to deduce that the following coefficients give the unique solution to the order conditions (15):

(16)

τ	$\tau_{3,1}$	$\tau_{3,2}$
$\frac{1}{2}$	$-\frac{1}{4}$	$\frac{1}{8}$
1	$-\frac{1}{2}$	$\frac{1}{4}$

With the coefficients (16) the EDRK method can be written as follows:

$$
\begin{aligned}
\mathbf{Y}_1 &= \mathbf{y}_n + \tfrac{1}{2}\, h\, F(\tau)(\mathbf{Y}_1) + \tfrac{1}{6}\, h^3 \left(-\tfrac{1}{4}\, F(\tau_{3,1})(\mathbf{Y}_1) + \tfrac{1}{8}\, F(\tau_{3,2})(\mathbf{Y}_1)\right) \\
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\, F(\tau)(\mathbf{Y}_1) + \tfrac{1}{6}\, h^3 \left(-\tfrac{1}{2}\, F(\tau_{3,1})(\mathbf{Y}_1) + \tfrac{1}{4}\, F(\tau_{3,2})(\mathbf{Y}_1)\right)\,.
\end{aligned}
\tag{17}
$$

Order	Tree	Order condition
1	τ	$\mathbf{b}(\tau)^T \mathbf{e} = 1$
2	$\tau_2 = [\tau]$	$\frac{1}{2}\mathbf{b}(\tau_2)^T \mathbf{e} + \mathbf{b}(\tau)^T \mathbf{a}(\tau)\mathbf{e} = \frac{1}{2}$
3	$\tau_{3,1} = [{}_2\tau]_2$	$\frac{1}{6}\mathbf{b}(\tau_{3,1})^T \mathbf{e} + \frac{1}{2}\mathbf{b}(\tau_2)^T \mathbf{a}(\tau)\mathbf{e}$ $+\frac{1}{2}\mathbf{b}(\tau)^T \mathbf{a}(\tau_2)\mathbf{e} + \mathbf{b}(\tau)^T \mathbf{a}(\tau)\mathbf{a}(\tau)\mathbf{e} = \frac{1}{6}$
	$\tau_{3,2} = [\tau^2]$	$\frac{1}{3}\mathbf{b}(\tau_{3,2})^T \mathbf{e} + \mathbf{b}(\tau_2)^T \mathbf{a}(\tau)\mathbf{e} + \mathbf{b}(\tau)^T (\mathbf{a}(\tau)\mathbf{e})^2 = \frac{1}{3}$
4	$\tau_{4,1} = [{}_3\tau]_3$	$\frac{1}{24}\mathbf{b}(\tau_{4,1})^T \mathbf{e} + \frac{1}{6}\mathbf{b}(\tau_{3,1})^T \mathbf{a}(\tau)\mathbf{e} + \frac{1}{4}\mathbf{b}(\tau_2)^T \mathbf{a}(\tau_2)\mathbf{e}$ $+\frac{1}{2}\mathbf{b}(\tau_2)^T \mathbf{a}(\tau)\mathbf{a}(\tau)\mathbf{e} + \frac{1}{6}\mathbf{b}(\tau)^T \mathbf{a}(\tau_{3,1})\mathbf{e}$ $+\frac{1}{2}\mathbf{b}(\tau)^T \mathbf{a}(\tau_2)\mathbf{a}(\tau)\mathbf{e} + \frac{1}{2}\mathbf{b}(\tau)^T \mathbf{a}(\tau)\mathbf{a}(\tau_2)\mathbf{e}$ $+\mathbf{b}(\tau)^T \mathbf{a}(\tau)\mathbf{a}(\tau)\mathbf{a}(\tau)\mathbf{e} = \frac{1}{24}$
	$\tau_{4,2} = [{}_2\tau^2]_2$	$\frac{1}{12}\mathbf{b}(\tau_{4,2})^T \mathbf{e} + \frac{1}{3}\mathbf{b}(\tau_{3,1})^T \mathbf{a}(\tau)\mathbf{e}$ $+\frac{1}{2}\mathbf{b}(\tau_2)^T (\mathbf{a}(\tau)\mathbf{e})^2 + \frac{1}{3}\mathbf{b}(\tau)^T \mathbf{a}(\tau_{3,2})\mathbf{e}$ $+\mathbf{b}(\tau)^T \mathbf{a}(\tau_2)\mathbf{a}(\tau)\mathbf{e} + \mathbf{b}(\tau)^T \mathbf{a}(\tau)(\mathbf{a}(\tau)\mathbf{e})^2 = \frac{1}{12}$
	$\tau_{4,3} = [\tau[\tau]]$	$\frac{1}{24}\mathbf{b}(\tau_{4,3})^T \mathbf{e} + \frac{1}{6}\mathbf{b}(\tau_{3,1})^T \mathbf{a}(\tau)\mathbf{e} + \frac{1}{3}\mathbf{b}(\tau_{3,2})^T \mathbf{a}(\tau)\mathbf{e}$ $+\frac{1}{4}\mathbf{b}(\tau_2)^T \mathbf{a}(\tau_2)\mathbf{e} + \frac{1}{2}\mathbf{b}(\tau_2)^T (\mathbf{a}(\tau)\mathbf{e})^2$ $+\frac{1}{2}\mathbf{b}(\tau_2)^T \mathbf{a}(\tau)\mathbf{a}(\tau)\mathbf{e} + \frac{1}{2}\mathbf{b}(\tau)^T ((\mathbf{a}(\tau)\mathbf{e})\ (\mathbf{a}(\tau_2)\mathbf{e}))$ $+\mathbf{b}(\tau)^T ((\mathbf{a}(\tau)\mathbf{e})\ (\mathbf{a}(\tau)\mathbf{a}(\tau)\mathbf{e})) = \frac{1}{8}$
	$\tau_{4,4} = [\tau^3]$	$\frac{1}{4}\mathbf{b}(\tau_{4,4})^T \mathbf{e} + \mathbf{b}(\tau_{3,2})^T \mathbf{a}(\tau)\mathbf{e}$ $+\frac{3}{2}\mathbf{b}(\tau_2)^T (\mathbf{a}(\tau)\mathbf{e})^2 + \mathbf{b}(\tau)^T (\mathbf{a}(\tau)\mathbf{e})^3 = \frac{1}{4}$

Table 5
Order conditions through order four for EDRK methods using tensor notation.

This method is implicit in the internal stage value $\mathbf{Y}_1$ which should be computed using Picard or Newton iteration. The iteration can be started from $\mathbf{Y}_1^{(0)} = \mathbf{y}_n$, but in general better estimates can be obtained using information from previous integration steps (see [7,8] for an overview).

It turns out that the method (17) has a number of desirable properties; it is symmetric and symplectic [16]. The method coefficients for the tree τ in (17) correspond to the implicit midpoint rule (one stage Gauss–Legendre implicit RK method) which has order two. The higher order trees in the EDRK method (17) serve to raise the order of the method to four. For comparison

an implicit RK method requires at least two stages to attain order four; the two stage Gauss–Legendre scheme of order four is given by:

$$\begin{array}{c|cc} \frac{1}{4} & \frac{1}{4} & \frac{1}{12}\left(3-2\sqrt{3}\right) \\ \frac{1}{12}\left(3+2\sqrt{3}\right) & \frac{1}{4} & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Thus EDRK methods in general attain higher order than RK methods for fewer stages, by making use of higher order elementary differentials instead of additional function evaluations.

5 Numerical example

In this section the EDRK method that was derived in Section 4.2 is applied to a differential system, for which analytic expressions for the elementary differentials are given.

5.1 Euler's equations

Euler's equations model the movement of a rigid body whose center of mass is fixed at the origin [14]:

$$\begin{aligned} {y_1}' &= a_1\, y_2\, y_3, \quad a_1 = \frac{I_2 - I_3}{I_2\, I_3} \\ {y_2}' &= a_2\, y_3\, y_1, \quad a_2 = \frac{I_3 - I_1}{I_3\, I_1} \\ {y_3}' &= a_3\, y_1\, y_2, \quad a_3 = \frac{I_1 - I_2}{I_1\, I_2} \end{aligned} \tag{18}$$

where the vector $\mathbf{y} = (y_1, y_2, y_3)^T$ represents the angular momentum in the body frame and I_1, I_2, I_3 are the principal moments of inertia. Two quadratic first integrals of the system are:

$$\begin{aligned} H(\mathbf{y}) &= \frac{1}{2}\left(\frac{{y_1}^2}{I_1} + \frac{{y_2}^2}{I_2} + \frac{{y_3}^2}{I_3}\right) \\ I(\mathbf{y}) &= {y_1}^2 + {y_2}^2 + {y_3}^2\,. \end{aligned}$$

The values of the principal moments are taken as $I_1 = 2$, $I_2 = 1$, $I_3 = 2/3$ and the initial conditions are $y_1 = \cos(11/10)$, $y_2 = 0$, $y_3 = \sin(11/10)$. With these initial values the invariant $I(\mathbf{y})$ has the effect of constraining the motion from $\mathbb{R}^3$ to the unit sphere. The invariant $H(\mathbf{y})$, in conjunction with $I(\mathbf{y})$, constrains the motion to an ellipsoid on the unit sphere.

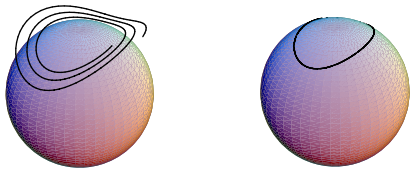

Fig. 7. Comparison of Euler's method (left) and the EDRK method (17) (right) for numerically solving Euler's equations (18).

5.2 Computation of Elementary Differentials

There are three elementary differentials involved in the method (17). For Euler's equations (18) these are given by:

$$F(\tau) = \begin{pmatrix} a_1\, y_2\, y_3 \\ a_2\, y_3\, y_1 \\ a_3\, y_1\, y_2 \end{pmatrix},$$

$$F(\tau_{3,1}) = \begin{pmatrix} a_1\, y_2\, y_3\ (2\, a_2\, a_3\, {y_1}^2 + a_1\, a_3\, {y_2}^2 + a_1\, a_2\, {y_3}^2) \\ a_2\, y_1\, y_3\ (a_2\, a_3\, {y_1}^2 + 2\, a_1\, a_3\, {y_2}^2 + a_1\, a_2\, {y_3}^2) \\ a_3\, y_1\, y_2\ (a_2\, a_3\, {y_1}^2 + a_1\, a_3\, {y_2}^2 + 2\, a_1\, a_2\, {y_3}^2) \end{pmatrix},$$

$$F(\tau_{3,2}) = \begin{pmatrix} 2\, a_1\, a_2\, a_3\, {y_1}^2\, y_2\, y_3 \\ 2\, a_1\, a_2\, a_3\, y_1\, {y_2}^2\, y_3 \\ 2\, a_1\, a_2\, a_3\, y_1\, y_2\, {y_3}^2 \end{pmatrix}.$$

Clearly there is a lot of repeated structure. An efficient implementation should exploit this by evaluating all of the elementary differentials simultaneously and sharing work across components.

5.3 Quadratic invariant conservation

In order to illustrate some of the desirable qualitative features of EDRK methods, we carried out a numerical experiment for Euler's equations (18) with step size $1/10$ on the interval $[0, 32]$. Picard iteration was used to solve the implicit equations with an error tolerance of one Unit in the Last Place in IEEE double precision. The results are depicted in Figure 7. It is known that Euler's method does not conserve quadratic invariants and the solution drifts away from the manifold (the unit sphere). In contrast the EDRK method (17) conserves both quadratic invariants to the order of unit roundoff and the solution remains on the manifold.

6 Conclusions

A modern graph theoretical framework for studying numerical methods for differential equations, based on rooted trees, has been reviewed. An extension to a new class of numerical methods has been outlined. The determination of order conditions is tedious and error prone when carried out by hand. This problem is compounded by the fact that the order conditions for the new schemes are a formal superset of those for Runge–Kutta methods and are therefore much more complicated. A *Mathematica* package has been developed to derive the new order conditions and details of the construction have been presented.

Symplectic numerical methods are advantageous for solving Hamiltonian systems of differential equations (see for example [17,8]). It is known that MDRK schemes, and hence Taylor Series methods, cannot be symplectic unless they are RK methods [9]. In contrast, it has recently been demonstrated that there exist symplectic EDRK schemes that are not RK methods. This result, together with simplified order conditions for symplectic EDRK methods are given in [16]. Thus by expanding the class of numerical integrators we are able to utilize derivatives of the vector field but maintain many of the desirable qualitative features that are commonplace in geometric numerical integration. A numerical example has been presented as illustration here and several more are given in [16].

Acknowledgements

The authors would like to thank Reinout Quispel and Per Christian Moan for collaboration on theoretical aspects of this work and the Mathematical and Statistical Sciences Department, LaTrobe University, Melbourne, for hospitality.

References

[1] Abramowitz, M., and Stegun, I. A., "Handbook of Mathematical Functions," 9th Ed., Dover Publications, New York, 1970.

[2] Butcher, J. C., *An algebraic theory of integration methods*, Math. of Comp. **26**, 117 (1972), 79–106.

[3] Butcher, J. C., "The Numerical Analysis of Ordinary Differential Equations," John Wiley and Sons, Chichester, 1987.

[4] Butcher, J. C., and Sanz-Serna, J. M., *The number of conditions for a Runge–Kutta method to have effective order p*, Appl. Numer. Math. **22** (1996), 103–111.

[5] Hairer, E., and Wanner, G., *On the Butcher group and general multi-value methods*, Computing **13** (1974), 1–15.

[6] Hairer, E., Nørsett, S. P., and Wanner, G., "Solving Ordinary Differential Equations I: Nonstiff Problems," 2nd Ed., Springer Series in Computational Mathematics **8**, Springer-Verlag, Berlin, 1993.

[7] Hairer, E., and Wanner, G., "Solving Ordinary Differential Equations II: Stiff and Differential Algebraic Problems," 2nd Ed., Springer Series in Computational Mathematics **14**, Springer-Verlag, Berlin, 1996.

[8] Hairer, E., Lubich, C., and Wanner, G., "Geometric Numerical Integration," Springer Series in Computational Mathematics **31**, Springer-Verlag, Berlin, 2002.

[9] Hairer, E., Murua, A., and Sanz-Serna, J. M., *The non-existence of symplectic multi-derivative Runge–Kutta methods*, BIT **34**, 1 (1994), 80–87.

[10] F. Harary, "Graph Theory," Addison–Wesley, London, 1969.

[11] Iserles, A., Ramaswami, G., and Sofroniou, M., *Runge–Kutta methods for quadratic ordinary differential equations*, BIT **38**, 2 (1998), 315–346.

[12] Kastlunger, K. H., and Wanner, G., *Runge Kutta processes with multiple nodes*, Computing (Arch. Elektron. Rechnen) **9** (1972), 9–24.

[13] Knuth, D. E., "The Art of Computer Programming: Seminumerical Algorithms," 3rd. Ed., Addison Wesley, Reading, Massachusetts, 1998.

[14] Marsden, J. E., and Ratiu, T. S., "Introduction to Mechanics and Symmetry," Springer-Verlag, New York, 1994.

[15] Merson, R. H., *An operational method for the study of integration processes*, Proc. Symp. Data Processing, Weapons Research Establishment, Salisbury, Australia, 110 (1957), 1–25.

[16] Moan, P. C., Quispel, R. G. W., Sofroniou, M., and Spaletta, G., *Symplectic elementary differential Runge–Kutta methods,* in preparation (2003).

[17] Sanz-Serna, J. M., and Calvo, M. P., *Numerical Hamiltonian Problems*, Applied Mathematics and Mathematical Computation **7**, Chapman and Hall, London, 1994.

[18] Sofroniou, M., *Symbolic derivation of Runge–Kutta methods*, J. Symb. Comp. **18**, 3 (1994), 265–296.

[19] Sofroniou, M., and W. Oevel, W., *Symplectic Runge–Kutta schemes I: order conditions*, SIAM J. Num. Anal. **34** (1997), 2063–2086.